Exercices d'informatique - Corrigé de la série n° 1

Cours 3INOC01

1. L'instruction *print*.
L'instruction **print**"..." affiche le texte entre guillemets. Sans virgule après les guillemets, l'affichage passe à la ligne.

2. Variables.
Les deux scripts produisent le même résultat. Dans le second, la chaîne de caractères est stockée dans une *variable*, nommée bernard. C'est la variable bernard qui est passée comme argument à la commande **print**. Une variable est une "case" de la mémoire dont le contenu peut changer. On accède au contenu de la case *via* le nom de la variable.

3. Variables.
Les deux scripts produisent le même résultat. Dans le second, le nombre est stocké dans une *variable*, nommée bernard. C'est la variable bernard qui est passée comme argument à la commande **print**. Une variable est une "case" de la mémoire dont le contenu peut changer. On accède au contenu de la case *via* le nom de la variable.

4. Types de nombres.
Les nombres de type `int` et `long` sont des nombres entiers. La taille des entiers de type `long` peut être aussi grande que la mémoire le permet. Les nombres de type `float` ont une précision finie comme le montre la différence entre n2−n3 et r2−r3.

5. Opérations sur les nombres.
Les opérations `*`, `**`, `/`, `//` et `%` correspondent respectivement à la multiplication, la division, la division avec troncature et le reste de la division. Pour les nombres de type `int` et `long` les opérations `/` et `//` ont le même effet.

6. Conversion de type.
La variable t1 est une chaîne et 2∗t1 est la concaténation de t1 avec elle-même.
La variable t2 est un entier et 2∗t2 donne simplement le double de 123, c'est-à-dire 246. L'instruction int permet de convertir une chaîne en nombre entier quand cela est possible. La présence d'une lettre dans la chaîne t1 produit une erreur.
L'instruction float permet de convertir un entier en réel.
L'instruction 't3' transforme le nombre t3 est chaîne. Ainsi, 2∗t4 est simplement la concaténation de la chaîne t4 avec elle-même, soit `123.0123.0`.

7. L'instruction *range*.
(1) L'instruction range(1,10,1) génère une liste de nombres 1, 2, 3, ⋯, 9
(2) (a) range(1,10,3)⟶ 1, 4, 7
(b) range(1,10,−1)⟶ ∅
(c) range(10,1,−1)⟶ 10, 9, 8, 7, ⋯ , 4, 3, 2
(d) range(10,1,1)⟶ ∅
(e) range(10,10,1)⟶ ∅

8. L'instruction *range*.

(1) `print range(1,50,3)`
(2) `print range(50,5,-4)`

9. Le type de données *list*.

Les objets de type *list* sont des listes. Les éléments d'une liste sont des objets quelconques et peuvent être de types différents.

10. Appartenance.

La commande *lettre* `in` *texte* retourne `False` si la *lettre* n'apparaît pas dans le *texte* et `True` sinon.

11. Test d'appartenance.

L'opérateur `x in l` teste si l'objet `x` est égal à l'un des éléments de la liste `l`. Il renvoie `True` en cas de succès et `False` sinon. L'opérateur `x not in l` est équivalent à `not (x in l)`.

12. L'instruction *for*.

Le script exécute 10 fois les deux instructions `print` et donc affiche dix fois le texte: "Bienvenue au cours d'informatique".

13. Boucles.

On peut, par exemple, utiliser les scripts suivants:

(1) pour afficher les nombres de 1 à 9

```
1 for n in range(1,10,1):
2     print n
```

(2) pour afficher les nombres 1, 0, 1, 4, 9, 16, 25, 36 et 49

```
1 for n in range(1,10,1):
2     a=(n-2)**2
3     print a
```

14. Boucles.

On peut, par exemple, utiliser le script

```
1 for lettre in "Bienvenue au cours d'informatique":
2     print lettre,"..."
```

15. Opérations sur les chaînes.

La chaîne `t1+t2` est la concaténation de `t1` et `t2` et la chaîne `t1*2` est la concaténation de la chaîne `t1` avec elle-même.

16. Variables.

La variable `gustave` vaut 27. La commande `gustave=gustave+5` n'est pas une équation ! (Remarquons que cette équation n'admet pas de solution !) C'est une opération d'**incrémentation**. Elle consiste à attribuer la valeur de `gustave+5` à la variable `gustave` (*i.e.* à remplacer le contenu de la variable `gustave` par le résultat de `gustave+5`). Ainsi, la variable `gustave` vaut 7 après l'exécution de la deuxième ligne, 12 après la troisième ligne, etc.

17. Variables.

La variable `a` vaut 16.

18. Variables.

La variable a vaut:

```
Bonjour Bonjour Bonjour Bonjour Bonjour Bonjour Bonjour Bonjour
```

19. Boucles.

On peut, par exemple, utiliser le script suivant:

```
1  n=0;
2  for lettre in "abcdefghijklmnopqrstuvwxyz":
3      n=n+1;
4      for k in range(1,n,1):
5          print lettre,
6      print lettre
```

ou plus simplement

```
1  n=0
2  for lettre in "abcdefghijklmnopqrstuvwxyz":
3      n=n+1
4      print n*lettre
```

20. Les instructions *if elif* et *else*.

Le script affiche le texte "Ce nombre est plus grand que 100." si n est strictement plus grand que 100, il affiche le texte "Ce nombre est plus petit que 10." si n$<$ 10 et il affiche le texte "Ce nombre est compris entre 10 et 100." dans tous les autres cas.

21. Boucles.

On peut, par exemple, utiliser le script suivant:

```
1  n=-10
2  for lettre in "ProgrammeravecPython!":
3      if n<=0:
4          m=1+(n+10)
5      else:
6          m=11-n
7      print (25-m)*" ",
8      print 2*m*lettre
9      n=n+1
```

22. Le module *turtle*.

Le module *turtle* (tortue en anglais) permet de réaliser simplement des dessins. Une tortue se déplace sur l'écran en laissant une trace ! Voici le rôle de chaque commande:

- **title**: donne un titre à la fenêtre.
- setup: détermine la taille de la fenêtre (width=largeur, height=hauteur). Si les nombres sont entre 0 et 1, la taille est donnée en % de la taille de l'écran, sinon les nombres correspondent à des pixels.
- **reset**: efface la fenêtre et place le curseur en son centre.
- forward: avance la tortue d'une longueur donnée en pixels. La direction par défaut de la tortue (après un **reset**) est à l'est.
- left / right: tourne la tortue d'un angle donné en degrés.
- delay: provoque une pause dans l'exécution du programme. Le temps est donné en millisecondes.
- position(): renvoie la position du curseur en pixels.
- write: écrit un texte à la position du curseur.

- color: change la couleur donnée soit en anglais soit par une combinaison de trois nombres entre 0 et 1.
- clear: efface le contenu de la fenêtre.
- fill (1)/ fill (0): active/désactive la fonction de remplissage.
- backward: recule le curseur d'une longueur donnée en pixels.
- up/down: lève (baisse) la tortue ce qui permet de la déplacer sans laisser de trace.
- goto: déplace le curseur jusqu'à une position x,y donnée en pixels.
- width: détermine la largeur du trait.
- circle: trace un cercle. Le rayon est donné en pixels.

23. L'instruction *list*.

L'instruction list ("texte") crée la liste `['t','e','x','t','e']`.

24. Eléments d'une liste.

Nous constatons que

(1) l[0] donne le premier élément de l,
(2) l[5] donne le sixième élément de l,
(3) l[−1] donne le dernier élément de l,
(4) l[−3] donne le 27^{e} élément de l,
(5) l[3:9:2] donne une tranche d'éléments de l partant du troisième élément jusqu'au huitième avec un pas de 2,
(6) l[3:] donne tous les éléments de l à partir du troisième,
(7) l[:−3] donne tous les éléments de l jusqu'au 27^{e},
(8) l[::2] donne tous les éléments de l ayant un indice pair,
(9) l[1:−1:2] donne tous les éléments de l ayant un indice impair,
(10) l[::−1] donne tous les éléments de l, mais dans l'ordre inverse.

25. Boucles et dessins.

(2) On peut, par exemple, utiliser le script suivant:

```
1 from turtle import *
2 couleurs=['red','blue','yellow','magenta','green']
3 for n in range(0,30,1):
4     color(couleurs[n%5])
5     forward(200-4*n)
6     left(90)
7 done()
```

26. Boucles et dessins.

On peut, par exemple, utiliser le script suivant:

```
1 from turtle import *
2 couleurs=('red','blue','yellow')
3 for n in xrange(1,30,2):
4     color(couleurs[n%3])
5     forward(10*n)
6     left(120)
7 done()
```

27. L'instruction *while*.

Le script affiche sur une ligne les chiffres 0 1 2 3 4 5 6 7 8 9. Sans la quatrième ligne, le script affiche des zéros tant (*while* en anglais) que n<10. Comme on a posé n=0 au début et que la valeur de n n'est pas modifiée dans le script, l'exécution ne s'arrête jamais. Pour interrompre le processus, il faut appuyer sur les touches `Ctrl` et `c` simultanément.

28. La fonction *input()*.

La fonction input() permet de stocker dans une variable un nombre donné par l'utilisateur. Si la ligne saisie par l'utilisateur n'est pas un nombre, le script ne s'exécute pas et l'interpréteur signale une erreur.

29. La fonction *raw_input()*.

La fonction raw_input() permet de stocker dans une variable un texte tapé au clavier par l'utilisateur.

30. Un petit jeu.

Le script suivant effectue ce qui est demandé.

```
1  print "Tapez un nombre:",
2  n=input()
3  while n!=3:
4      print "NON"
5      print "Tapez un nombre:",
6      n=input()
7  print "OUI"
```

31. Un petit jeu.

Le script suivant effectue ce qui est demandé.

```
1  print "Tapez un nombre:",
2  n=input()
3  while n%2!=0:
4      print "Ce nombre est impair"
5      print "Tapez un nombre:",
6      n=input()
7  print "Ce nombre est pair"
```

32. L'instruction *break*.

L'instruction **break** ne peut apparaître que dans le corps d'une boucle: lorsqu'elle est exécutée, elle met fin à cette boucle.

33. Jeu du chaud-froid.

On peut utiliser le script suivant:

```
1  print "Devinez le nombre qui se cache ... !"
2  m=15
3  while True:
4      print "Tapez un nombre: ",
5      n=input()
6      if n==m:
7          print "Bravo !"
8          break
9      elif abs(n-m)>20:
10         print "C'est glacial !"
11     elif abs(n-m)<=20 and abs(n-m)>10:
12         print "C'est froid !"
13     elif abs(n-m)<=10:
14         print "C'est presque chaud !"
```